EXHIBIT 13

AMBULANT: A Fast, Multi-Platform Open Source SMIL Player

Dick C.A. Bulterman, Jack Jansen, Kleanthis Kleanthous, Kees Blom and Daniel Benden
CWI: Centrum voor Wiskunde en Informatica
Kruislaan 413
1098 SJ Amsterdam, The Netherlands
+31 20 592 43 00
Dick.Bulterman@cwi.nl

ABSTRACT

This paper provides an overview of the Ambulant Open SMIL player. Unlike other SMIL implementations, the Ambulant Player is a reconfigureable SMIL engine that can be customized for use as an experimental media player core. The Ambulant Player is a reference SMIL engine that can be integrated in a wide variety of media player projects. This paper starts with an overview of our motivations for creating a new SMIL engine, then discusses the architecture of the Ambulant Core (including the scalability and custom integration features of the player). We close with a discussion of our implementation experiences with Ambulant instances for Windows, Mac and Linux versions for desktop and PDA devices.

Categories and Subject Descriptors

H.5.1 Multimedia Information Systems [Evaluation],
H.5.4 Hypertext/Hypermedia [Navigation].

General Terms

Experimentation, Performance, Verification

Keywords

SMIL, Player, Open-Source, Demos

1. MOTIVATION

The Ambulant Open SMIL Player is an open-source, full featured SMIL 2.0 player. It is intended to be used within the researcher community (in and outside our institute) in projects that need source code access to a production-quality SMIL player environment. It may also be used as a stand-alone SMIL player for applications that do not need proprietary media formats. The player supports a range of SMIL 2.0 profiles (including desktop and mobile configurations) and is available in distributions for Linux, Macintosh, and Windows systems ranging from desktop devices to PDA and handheld computers.

While several SMIL player implementations exist, including the RealPlayer [4], InternetExplorer [5], PocketSMIL [7], GR*i*NS [6], X-SMILES [8] and various proprietary implementations for mobile devices, we developed Ambulant for three reasons:

MM'04, October 10-16, 2004, New York, New York, USA.

- None of the existing SMIL players provides a complete and correct SMIL 2.0 implementation. The Ambulant player implements all of SMIL, based on the SMIL 2.0 Language profile plus extensions to support advanced animation and the needs of the mobile variant used by the 3GPP/PSS-6 SMIL specification [9].
- All commercial SMIL players are geared to the presentation of proprietary media. The Ambulant player uses open-source media codecs and open-source network transfer protocols, so that the player can be easily customized for use in a wide range of research projects.
- Our goal is to build a platform that will encourage the development of comparable multimedia research output. By providing what we expect will be a standard baseline player, other researchers and development organizations can concentrate on integrating extensions to the basic player (either in terms of new media codecs or new network control algorithms). These extensions can then be shared by others.

In contrast to the Helix client architecture [10], which also moved to a GPL core in mid-2004, the Ambulant player supports a wider range of SMIL target application architectures, it provides a more complete and correct implementation of the SMIL language, it provides much better performance on low-resource devices and it provides a more extensible media player architecture. It also provides an implementation that includes all of the media codecs as part of the open client infrastructure.

The Ambulant target community is not viewers of media content, but developers of multimedia infrastructures, protocols and networks. Our goal has been to augument the existing partial SMIL implementations produced by many groups with a complete implementation that supports even the exotic features of the SMIL language.

The following sections provide an introduction to the architecture of the player and describe the state of the various Ambulant implementations. We then discuss how the Ambulant Core can be re-purposed in other projects. We start with a discussion of Ambulant's functional support for SMIL.

2. FUNCTIONAL SUPPORT FOR SMIL 2.0

The SMIL 2.0 recommendation [1] defines 10 functional groups that are used to structure the standard's 50+ modules. These modules define the approximately 30 XML *elements* and 150 *attributes* that make up the SMIL 2.0 language. In addition to defining modules, the SMIL 2.0 specification also defines a number of SMIL *profiles*: collection of elements, attributes and attribute values that are targeted to meet the needs of a particular implementation community. Common profiles include the full SMIL 2.0 Language, SMIL Basic, 3GPP SMIL, XHTML+SMIL and SMIL 1.0 profiles.

A review of these profiles is beyond the scope of this paper (see [2]), but a key concern of Ambulant's development has been to provide a player core that can be used to support a wide range of SMIL target profiles with custom player components. This has resulted in an architecture that allows nearly all aspects of the player to be plug-replaceable via open interfaces. In this way, tailored layout, scheduling, media processing and interaction modules can be configured to meet the needs of individual profile requirements. The Ambulant player is the only player that supports this architecture.

The Ambulant player provides a direct implementation of the SMIL 2.0 Language profile, plus extensions that provide enhanced support for animation and timing control. Compared with other commercial and non-commercial players, the Ambulant player implements not only a core scheduling engine, it also provides complete support for SMIL layout, interaction, content control and networking facilities. Ambulant provides the most complete implementation of the SMIL language available to date.

3. AMBULANT ARCHITECTURE

This section provides an overview of the architecture of the Ambulant core. While this discussion is high-level, it will provide sufficient detail to demonstrate the applicability of Ambulant to a wide range of projects. The sections below consider the high-level interface structure, the common services layer and the player common core architecture.

3.1 The High-Level Interface Structure

Figure 1 shows the highest level player abstraction. The player core support top-level control external entry points (including play/stop/pause) and in turn manages a collection of external factories that provide interfaces to data sources (both for standard and pseudo-media), GUI and window system interfaces and interfaces to renderers. Unlike other players that treat SMIL as a datatype [4],[10], the Ambulant engine has a central role in interaction with the input/output/screen/devices interfaces.

This architecture allows the types of entry points (and the moment of evaluation) to be customized and separated from the various data-sources and renderers. This is important for integration with environments that may use non-SMIL layout or special device interface processing.

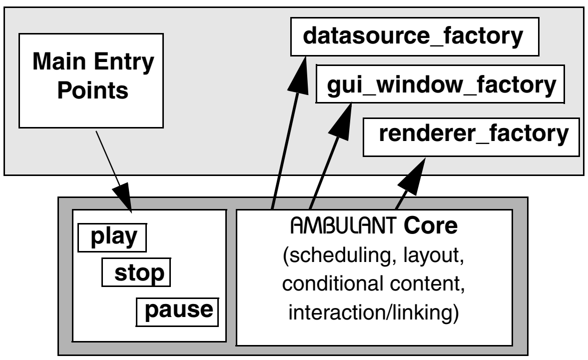

Figure 1. Ambulant high-level structure.

3.2 The Common Services Layer

Figure 2 shows a set of common services that are supplied for the player to operate. These include operating systems interfaces, drawing systems interfaces and support for baseline XML functions.

All of these services are provided by Ambulant; they may also be integrated into other player-related projects or they may be replaced by new service components that are optimized for particular devices or algorithms.

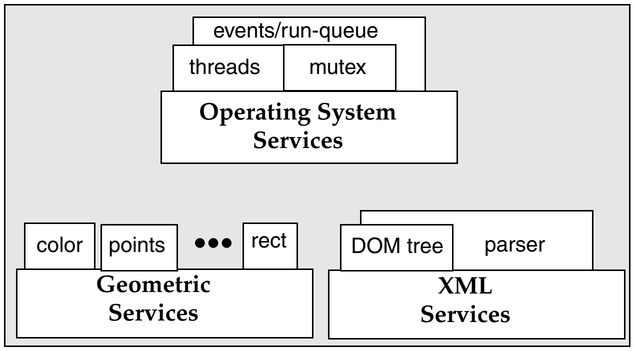

Figure 2. Ambulant Common Services Layer.

3.3 The Player Common Core

Figure 3 shows a slightly abstracted view of the Ambulant common core architecture. The view is essentially that of a single instance of the Ambulant player. Although only one class object is shown for each service, multiple interchangeable implementations have been developed for all objects (except the DOM tree) during the player's development. As an example, multiple schedulers have been developed to match the functional capabilities of various SMIL profiles.

Arrows in the figure denote that one abstract class depends on the services offered by the other abstract class. Stacked boxes denote that a single instance of the player will contain instances of multiple concrete classes implementing that abstract class: one for audio, one for images, etc. All of the stacked-box abstract classes come with a factory function to create the instances of the required concrete class.

The bulk of the player implementation is architected to be platform independent. As we will discuss, this platform independent component has already been reused for five separate player implementations. The platform dependent portions of the player include support for actual rendering, UI interaction and datasource processing and control.

When the player is active, there is a single instance of the scheduler and layout manager, both of which depend on the DOM tree object. Multiple instances of data source and playable objects are created. These interact with multiple abstract rendering surfaces. The playable abstract class is the scheduler interface (play, stop) for a media node, while the renderer abstract class is the drawing interface (redraw). Note that not all playables are renderers (audio, SMIL animation).

The architecture has been designed to have all components be replaceable, both in terms of an alternative implementation of a given set of functionality and in terms of a complete re-

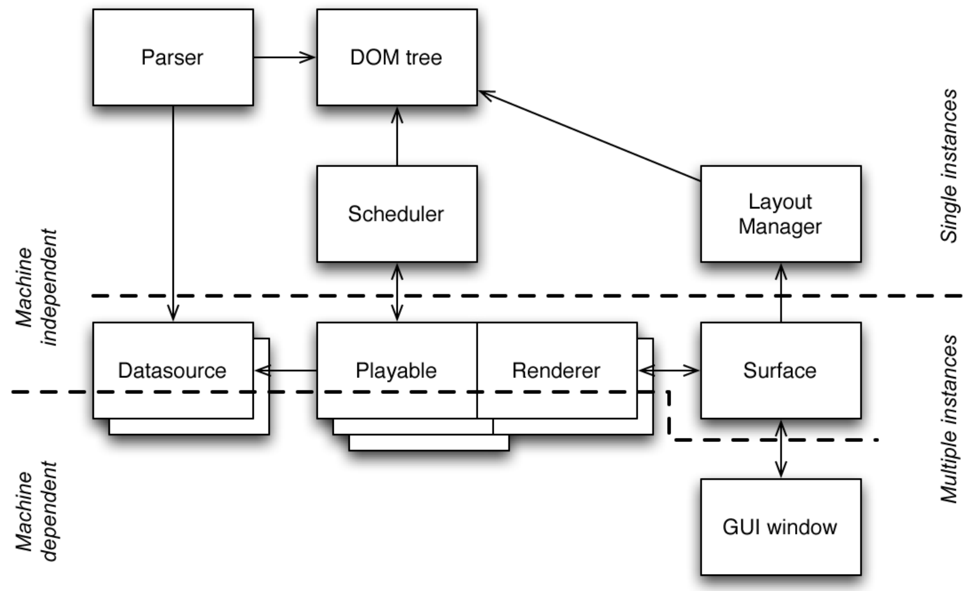

Figure 3. Ambulant Core Architecture.

purposing of the player components. In this way, the Ambulant core can be migrated to being a special purpose SMIL engine or a non-SMIL engine (such as support for MPEG-4 or other standards).

The abstract interfaces provided by the player do not require a “SMIL on Top” model of document processing. The abstract interface can be used with other high-level control models (such as in an XHTML+SMIL implementation), or to control non-SMIL lower-level rendering (such as timed text).

Note that in order to improve readability of the illustration, all auxiliary classes (threading, geometry and color handling, etc.) and several classes that were not important for general understanding (player driver engine, transitions, etc.) have been left out of the diagram.

4. IMPLEMENTATION EXPERIENCES

This section will briefly review our implementation experiences with the Ambulant player. We discuss the implementation platforms used during SMIL’s development and describe a set of test documents that were created to test the functionality of the Ambulant player core. We conclude with a discussion on the performance of the Ambulant player.

4.1 Implementation Platforms

SMIL profiles have been defined for a wide range of platforms and devices, ranging from desktop implementations to mobile devices. In order to support our research on distributed SMIL document extensions and to provide a player that was useful for other research efforts, we decided to provide a wide range of SMIL implementations for the Ambulant project.

The Ambulant core is available as a single C++ source distribution that provides support for the following platforms:

- *Linux*: our source distribution include makefiles that are used with the RH-8 distribution of Linux. We provide support for media using the FF-MPEG suite [11]. The player interface is built using the Qt toolkit [12].
- *Macintosh*: Ambulant supports Mac OS X 10.3. Media rendering support is available via the internal Quicktime API and via FF-MPEG. The player user interface uses standard Mac conventions and support (Coca).
- *Windows*: Ambulant provides conventional Win32 support for current generation Windows platforms. It has been most extensively tested with XP (Home, Professional and TabletPC) and Windows-2000. Media rendering include third-party and local support for imaging and continuous media. Networking and user interface support are provided using platform-embedded libraries.
- *PocketPC*: Ambulant supports PocketPC-2000, PocketPC-2002 and Windows Mobile 2003 systems. The PocketPC implementations provide support for basic imaging, audio and text facilities.
- *Linux PDA support*: Ambulant provides support for the Zaurus Linux-PDA. Media support is provided via the FF-MPEG library and UI support is provide via Qt. Media support includes audio, images and simple text.

In each of these implementations, our initial focus has been on providing support for SMIL scheduling and control functions. We have not optimized media renderer support in the Ambulant 1.0 releases, but expect to provide enhanced support in future versions.

4.2 Demos and Test Suites

In order to validate the Ambulant player implementation beyond that available with the standard SMIL test suite [3], several demo and test documents have been distributed with the player core. The principal demos include:

- *Welcome*: A short presentation that exercises basic timing, media rendering, transformations and animation.

- *NYC*: a short slideshow in desktop and mobile configurations that exercises scheduling, transformation and media rendering.
- *News*: a complex interactive news document that tests linking, event-based activation, advanced layout, timing and media integration. Like NYC, this demo support differentiated mobile and desktop configurations.
- *Links*: a suite of linking and interaction test cases.
- *Flashlight*: an interactive user's guide that tests presentation customization using custom test attributes and linking/interaction support.

These and other demos are distributed as part of the Ambulant player web site [13].

4.3 Performance Evaluation

The goal of the Ambulant implementation was to provide a complete and fast SMIL player. We used a C++ implementation core instead of Java or Python because our experience had shown that on small devices (which we feel hold significant interest for future research), the efficiency of the implementation still plays a dominant role. Our goal was to be able to read, parse, model and schedule a 300-node news presentation in less than two seconds on desktop and mobile platforms. This goal was achieved for all of the target platforms used in the player project. By comparison, the same presentation on the Oratrix GR*i*NS PocketPC player took 28 seconds to read, parse and schedule. (The Real PocketPC SMIL player and the PocketSMIL players were not able to parse and schedule the document at all because of their limited SMIL language support.)

In terms of SMIL language performance, our goal was to provide a complete implementation of the SMIL 2.0 Language profile [14]. Where other players have implemented subsets of this profile, Ambulant has managed to implement the entire SMIL 2.0 feature set with two exceptions: first, we currently do not support the prefetch elements of the content control modules; second, we provide only single top-level window support in the platform-dependent player interfaces. Prefetch was not supported because of the close association of an implementation with a given streaming architecture. The use of multiple top-level windows, while supported in our other SMIL implementation, was not included in version 1.0 of Ambulant because of pending working on multi-screen mobile devices. Both of these feature are expected to be supported in the next release of Ambulant.

5. CURRENT STATUS AND AVAILABILITY

This paper describes version 1.0 of the Ambulant player, which was released on July 12, 2004. (This version is also known as the Ambulant/O release of the player.) Feature releases and platform tuning are expected to occur in the summer of 2004. The current release of Ambulant is always available via our SourceForge links [13], along with pointers to the most recent demonstrators and test suites.

The W3C started its SMIL 2.1 standardization in May, 2004. At the same time, the W3C's timed text working group is completing its first public working draft. We will support both of these activities in upcoming Ambulant releases.

6. CONCLUSIONS

While SMIL support is becoming ubiquitous (in no small part due to its acceptance within the mobile community), the availability of open-source SMIL players has been limited. This has meant that any group wishing to investigate multimedia extensions or high-/low-level user or rendering support has had to make a considerable investment in developing a core SMIL engine.

We expect that by providing a high-performance, high-quality and complete SMIL implementation in an open environment, both our own research and the research agendas of others can be served. By providing a flexible player framework, extensions from new user interfaces to new rendering engines or content control infrastructures can be easily supported.

7. ACKNOWLEDGEMENTS

This work was supported by the Stichting NLnet in Amsterdam.

8. REFERENCES

[1] W3C, SMIL Specification, http://w3.org/AudioVideo.

[2] Bulterman, D.C.A. and Rutledge, L., SMIL 2.0: Interactive Multimedia for Web and Mobile Devices, Springer, 2004.

[3] W3C, SMIL 2.0 Standard Testsuite, //www.w3.org/2001/SMIL20/testsuite/

[4] RealNetworks, The RealPlayer 10, www.real.com/

[5] Microsoft, HTML+Time in Internet Explorer 6, msdn.microsoft.com/workshop/author/behaviors/time.asp

[6] Oratrix, The GRiNS 2.0 SMIL Player. www.oratrix.com/

[7] INRIA, The PocketSMIL 2.0 Player, wam.inrialpes.fr/software/pocketsmil/.

[8] X-SMILES.org, X-SMILES: An Open XML-Browser for Exotic Applications.www.xsmiles.org/

[9] 3GPP Consortium, The Third-Generation Partnership Project (3GPP) SMIL PSS-6 Profile. www.3gpp.org/ftp/Specs/ archive/26_series/26.246/ 26246-003.zip

[10] Helix Community, The Helix Player. helixcommunity.org/.

[11] FFMPEG, FF-MPEG: A Complete Solution for Recording, Converting and Streaming Audio and Video. ffmpeg.sourceforge.net/

[12] Trolltech, Qtopia: The QT Palmtop Environment. qpe.sourceforge.net/

[13] Ambulant Project, The Ambulant 1.0 Open Source SMIL 2.0 Player, www.ambulantPlayer.org/.

[14] Bulterman, D.C.A., A Linking and Interaction Evaluation Test Set for SMIL, Proc. ACM Hypertext 2004, Santa Cruz, August, 2004.